IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Harish G. PATIL et al.	§	Confirmation No.:	7608
		§		
Serial No.:	09/723,687	§	Group Art Unit:	2183
		§		
Filed:	11/28/2000	§	Examiner:	Aimee J. Li
		§		
For:	Branch Prediction Combining Static And Dynamic Prediction Techniques	§ § § §	Docket No.:	200308341-1

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Date: December 31, 2007

Sir:

Appellants hereby submit this Appeal Brief in connection with the above-identified application. A Notice of Appeal was electronically filed on November 7, 2007.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company (HPDC), a Texas Limited Partnership, having its principal place of business in Houston, Texas. HPDC is a wholly owned affiliate of Hewlett-Packard Company (HPC). HPC merged with Compaq Computer Corporation (CCC) which owned Compaq Information Technologies Group, L.P. (CITG). The Assignment from the inventors to CCC was recorded on November 28, 2000, at Reel/Frame 011304/0641. The Assignment from CCC to CITG was recorded on January 16, 2002, at Reel/Frame 012394/0118. The Change of Name document was recorded on May 12, 2004, at Reel/Frame 014628/0103.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1-22.
Claim cancellations:	2, 3, 10, 11, 14, 19 and 20.
Added claims:	None.
Presently pending claims:	1, 4-9, 12, 13, 15-18, 21 and 22.
Presently appealed claims:	1, 4-9, 12, 13, 15-18, 21 and 22.

IV. STATUS OF THE AMENDMENTS

No claims were amended after the final Office Action dated October 5, 2007.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Microprocessors fetch instructions from memory and execute the instructions. Some microprocessors have a pipelined architecture comprising various stages of processing (e.g., fetching, decoding, scheduling, executing, etc.). As such, multiple instructions can be processed through the pipeline in an efficient manner. For example, while one instruction is being scheduled, the next instruction can be fetched and decoded. Appellants' disclosure, pages 1-2.

A conditional branch instruction is an instruction that includes a condition that can be true or false. When the instruction is executed, the condition is checked to determine whether it is true or false. If the condition is true, program control jumps to a predefined set of instructions (the branch is said to be "taken"), whereas if the condition is false, control may continue with the instructions immediately following the conditional branch instruction (the branch is said to be "not taken"). Appellants' disclosure, page 2.

In this scenario, a problem occurs as to which instructions to fetch and place into the pipeline following a conditional branch instruction when the outcome of the instruction is not yet known. Appellants' disclosure, pages 2-3. To solve this problem, some microprocessors include a hardware-based branch predictor. Numerous variations of branch predictors are possible, but, in general, a branch predictor keeps a history of the previous executions of each conditional branch instruction and makes a prediction "on-the-fly" about a future execution of the branch instruction based on the historical information. For example, if the previous execution of a branch resulted in the condition being true, then the branch predictor may predict that the condition will be true the next time the branch instruction is fetched. The prediction may turn out to be correct or incorrect. "The branch predictor merely predicts the future outcome of the conditional branch instruction; the true outcome will not be accurately known until the branch instruction is actually executed. If the branch predictor turns out to have made the correct prediction, then instructions that must be executed are already in the pipeline. If the prediction turns out to have been inaccurate, then the incorrect instructions that had been fetched must be thrown out and the

correct instructions fetched. Performance suffers on mispredictions and increases on correct predictions. Choosing a branch prediction scheme that results in correct predictions much more often than mispredictions will result in the performance increase gained from correct predictions outweighing the performance hit on mispredictions." Appellants' disclosure, page 3.

"Aliasing" is a problem symptomatic of many dynamic, hardware-based branch predictors. Aliasing refers to the problem in which multiple branch instructions in a program may index to a common location in a table of predictions. Aliasing undesirably can cause the predictions for one branch instruction to be influenced by the outcomes of another branch instruction that indexes to the same location in a prediction table. Appellants' disclosure, page 4.

Appellants have addressed this issue by making use of static software branch prediction instructions such as instruction 208 in Figure 2. As disclosed by Appellants, such instructions comprise multiple groups of prediction bits. Each group is configurable to provide prediction information for a separate conditional branch instruction, such as any one or more of instructions 201-207. Thus, Appellants' static branch prediction instructions contain prediction information about more than one conditional branch instruction. None of the art of record teaches such a static branch prediction instruction.

The invention of claim 1 is directed to a computer system that comprises a processor[1] which in turn comprises a hardware branch predictor.[2] In the system, certain software instructions are executed by the processor. The software instructions comprise at least conditional branch instructions[3] and separate static branch prediction instructions.[4] The static branch prediction instructions comprise a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch

---

[1] Fig. 1, processor 100.

[2] Fig. 1, processor 102. See also p. 8, lines 2-5.

[3] Fig. 2, one or more of instructions 201-207.

[4] Fig. 2, instruction 208. See also page p. 8, lines 10-12; p. 9, line 21; p. 11, lines 19-21.

instruction.[5] The processor predicts one or more condition branch instructions by executing the static branch prediction instructions.

The invention of claim 9 comprises fetch logic[6] that fetches program instructions from a source external to the processor.[7] The processor also comprises a dynamic branch predictor[8] that supplies predictions regarding conditional branch instructions to the fetch logic. An instruction queue[9] is also provided. The fetch logic stores fetched instructions in the instruction queue. An execution unit executes instructions provided from the instruction queue.[10] The fetch logic examines fetched instructions for a predetermined register identifier that identifies that instruction as a static branch prediction instruction. Such a static branch instruction provides separate static branch prediction information about a plurality of conditional branch instructions.[11]

The invention of claim 18 is directed to a method of predicting the outcome of conditional branch instructions.[12] The method comprises including a static branch predictor software instruction[13] in a program. The branch prediction software instruction includes branch prediction information configurable to pertain to a plurality of conditional branch instructions in the program.[14] The method further includes fetching branch prediction software instructions and decoding the branch prediction software instructions to determine if the decoded instruction is a branch prediction software instruction. If the decoded instruction is a branch prediction software instruction, then the method continues by predicting at least

[5] Fig. 2. See also p. 13, line 3 through p. 15, line 18.

[6] Fig. 1 processor 101.

[7] See p. 8, lines 13-17.

[8] Fig. 1, processor 102. See also p. 8, lines 2-5.

[9] Fig. 1 processor 106.

[10] Fig. 1, 114. See p. 9, lines11-14.

[11] Fig. 2. See also p. 13, line 3 through p. 15, line 18.

[12] Fig. 2, one or more of instructions 201-207. See also p. 6, lines 8-22.

[13] Fig. 2, instruction 208.

[14] Fig. 2. See also p. 13, line 3 through p. 15, line 18.

one conditional branch instructing based on the branch prediction information. If, however, the decoded instruction is not a branch prediction software instruction, then the decoded instruction is executed.[15]

[15] See p. 6, lines 8-22; p. 14 line 20 through p. 15, line 18.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1, 4-9, 12, 13, 15-18, 21 and 22 are obvious Mallick (U.S. Pat. No. 5,752,014) in view of Intel's *Intel® IA-64 Architecture Software Developer's Manual Volume 1:IA-64 Application Architecture*, (hereinafter referred to as "Intel Volume 1") and further in view of Blaner (U.S. Pat. No. 5,649,178).

VII. ARGUMENT

A. Overview of Mallick

As correctly noted by the Examiner, Mallick discloses a hardware dynamic branch predictor. See e.g., Figures 1 and 2. "The branch processing unit includes…a branch prediction unit for predicting the resolution of a conditional branch instruction utilizing the selected branch prediction methodology." See Abstract. The Examiner also correctly concedes that Mallick does not disclose the use of branch prediction instructions.

B. Overview of Intel Volume 1

Intel Volume 1 discloses the use of "separate Branch Prediction instructions (brp) where the entire instruction is hint information." Page 4-29. Each branch prediction instruction includes prediction information relative to **only one** conditional branch instruction. A displacement value is included in the branch prediction instruction to identify the particular conditional branch instruction referenced by the prediction information. Page 4-30.

C. Overview of Blaner

Blaner provides an improvement to hardware-based conditional branch instructions predictors. The Background section of the Blaner patent explains that once a conditional branch instruction is initially executed, a history of execution of that instruction is established which is used to predict the result of subsequent executions of the same instructions. Various hardware branch prediction schemes exist. "One such scheme involves dynamic prediction of branch outcomes by tagging branch instructions in a cache with predictive information regarding their outcomes." The problem Blaner addresses is the loss of the predictive information when a cache line containing a conditional branch instruction is evicted from the cache, as is the nature of cache operation. If the evicted branch instruction is ever again brought back into the cache, establishing the initial state of the instruction's predictive information is problematic. See col. 1, line 48 through col. 2, line 34.

As noted above, Blaner is directed to hardware, dynamic branch prediction logic. More specifically and with reference to Figure 1, the "[c]ache/BHT[16] 102 provides temporary storage for lines of data received from memory 100 and also provides branch history bits for branch instructions contained within the data. Thus, cache/BHT 102 provides for dynamic prediction of branch outcomes by tagging branch instructions in the cache with predictive information regarding their outcomes... ." Co. 3, lines 10-18.

Blaner's solution to his stated problem is the use of the "branch history cache" 103 depicted in Figure 1. Blaner's branch history cache 103 provides storage for evicted predictive information. As such, if a line containing a branch instruction is evicted from the cache, the associated predictive information tagged to that cache line is off-loaded to the branch history cache 103. Then, if that branch instruction is subsequently brought back into the cache, the instruction's predictive information is retrieved from the branch history cache 103 as well. See e.g., col. 2, lines 38-47 and col. 4, lines 4-21. Blaner thus provides an improvement to dynamic, hardware-based branch predictors.

D. Claims 1 and 4-8

Claim 1 requires, among other features, "static branch prediction instructions." As claimed, each such static branch prediction instruction comprises "a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch instruction." The Examiner correctly concedes that Mallick does not teach separate static branch prediction instructions, and instead turned the combination of Intel Volume 1 and Blaner.

Intel Volume 1, however, does not teach a branch prediction instruction that comprises a group of prediction bits in which each group is configurable to provide prediction information for a separate branch instruction. At most, Intel Volume 1 discloses a branch prediction instruction that provides prediction information for **only one** branch conditional instruction.

---

[16] "BHT" stands for Branch History Table. Col. 3, lines 9-10.

Blaner is also deficient in this regard. Blaner does not teach or suggest a static branch prediction instruction that comprises "a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch instruction." Blaner does not even teach or suggest the use of processor-executed static branch prediction instructions. Instead, Blaner teaches the use of dynamic branch prediction hardware.

In the most recent Final Office Action, the Examiner acknowledged that Intel Volume 1's static branch prediction instruction provides prediction information about only a single conditional branch instruction. The Examiner also stated that this deficiency of Intel Volume 1 is satisfied by Blaner which, according to the Examiner, teaches hardware-based branch prediction information about multiple conditional branch instructions. However, the fact remains that prior art references teaches a static branch prediction instruction as specified in claim 1, that is, an instruction which comprises prediction information about multiple conditional branch instructions. Absent the teachings provided by Appellants themselves, which is not permitted in an obviousness analysis, none of the art teaches or suggests the particular kind of static branch prediction software instruction specified in claim 1.

At least for these reasons, the Examiner erred in rejecting claim 1 and its associated dependent claims.

E. Claims 9 and 12-17

Claim 9 refers to a static branch prediction instruction that provides "separate static branch prediction information about a plurality of conditional branch instructions." As explained above, the Examiner conceded that Mallick lacks this feature, and Intel Volume 1 and Blaner are deficient as well. For at least this reason, the Examiner erred in rejecting claim 9 and associated dependent claims.

F. Claims 18, 21, and 22

Claim 18 is directed to a method that requires, among other actions, "including a static branch predictor software instruction in a program, said branch

prediction software instruction including branch prediction information configurable to pertain to a plurality of conditional branch instructions in the program." As explained above, the Examiner conceded that Mallick lacks this feature, and Intel Volume 1 and Blaner are deficient as well. For at least this reason, the Examiner erred in rejecting claim 9 and associated dependent claims.

G. Conclusion

For the reasons stated above, Appellants respectfully submit that the Examiner erred in rejecting all pending claims. It is believed that no extensions of time or fees are required, beyond those that may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required (including fees for net addition of claims) are hereby authorized to be charged to Hewlett-Packard Development Company's Deposit Account No. 08-2025.

Respectfully submitted,

/Jonathan M. Harris/

Jonathan M. Harris
PTO Reg. No. 44,144
CONLEY ROSE, P.C.
(713) 238-8000 (Phone)
(713) 238-8008 (Fax)
ATTORNEY FOR APPELLANTS

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
Legal Dept., M/S 35
P.O. Box 272400
Fort Collins, CO 80527-2400

VIII. CLAIMS APPENDIX

1. (Previously presented) A computer system, comprising:

a processor that comprises a hardware branch predictor; and

software instructions executed by said processor, said software instructions comprising conditional branch instructions and separate static branch prediction instructions;

said static branch prediction instructions comprise a plurality of groups of static branch prediction bits, each group being configurable to provide prediction information for a separate conditional branch instruction; and

wherein said processor predicts one or more condition branch instructions by executing said static branch prediction instructions.

2. (Canceled).

3. (Canceled).

4. (Previously presented) The computer system of claim 1, wherein each group of static branch prediction bits comprises a pair of bits.

5. (Previously presented) The computer system of claim 1 wherein said prediction information comprises a member selected from the group consisting of: do not use static prediction, predict taken, and predict not taken.

6. (Original) The computer system of claim 4 wherein each pair of prediction bits corresponds to another instruction and each pair of prediction bits is encoded as: 00 and 01 mean do not use static prediction, 10 means predict taken and 11 means predict not taken.

7. (Previously presented) The computer system of claim 1 wherein said static branch prediction bits comprise static branch prediction information that

comprises encoded information directing the processor to ignore the predictions supplied by the hardware branch predictor.

8. (Previously presented) The computer system of claim 1 wherein said hardware branch predictor comprises a log in which the results of all executed conditional branch instructions are stored.

9. (Previously presented) A processor, comprising:

fetch logic that fetches program instructions from a source external to said processor;

a dynamic branch predictor coupled to said fetch logic, said dynamic branch predictor supplies predictions regarding conditional branch instructions to said fetch logic;

an instruction queue coupled to said dynamic predictor, said fetch logic storing fetched instructions in said instruction queue; and

an execution unit coupled to said instruction queue and executing instructions provided from said instruction queue;

said fetch logic examines fetched instructions for a predetermined register identifier that identifies that instruction as a static branch prediction instruction that provides separate static branch prediction information about a plurality of conditional branch instructions.

10. (Canceled).

11. (Canceled).

12. (Previously presented) The computer system of claim 10, wherein said separate static branch prediction information for each conditional branch instruction comprises a pair of bits.

13. (Previously presented) The processor of claim 9 wherein said prediction information comprises a member selected from the group consisting of: do not use static prediction, predict taken, and predict not taken.

14. (Canceled).

15. (Previously presented) The processor of claim 9 wherein said static branch prediction instruction comprises branch prediction bits that directs said fetch logic to ignore the predictions supplied by the dynamic branch predictor.

16. (Previously presented) The processor of claim 9 wherein said dynamic branch predictor comprises a log in which the results of all executed conditional branch instructions are stored.

17. (Original) The processor of claim 9 wherein said predetermined identifier comprises a register identifier.

18. (Previously presented) A method of predicting the outcome of conditional branch instructions, comprising:

including a static branch predictor software instruction in a program, said branch prediction software instruction including branch prediction information configurable to pertain to a plurality of conditional branch instructions in the program;

fetching said branch prediction software instructions;

decoding said branch prediction software instructions to determine if said decoded instruction is a branch prediction software instruction; and

if said decoded instruction is a branch prediction software instruction, then predicting at least one conditional branch instructing based on said branch prediction information; and

if said decoded instruction is not a branch prediction software instruction, then executing said decoded instruction.

19. (Canceled).

20. (Canceled).

21. (Previously presented) The method of claim 18 wherein said branch prediction information comprises pairs of bits, each pair corresponding to another instruction.

22. (Previously presented) The method of claim 21 further comprising decoding said pairs of bits to determine whether, for said other instruction corresponding to said pair, said other instruction is predicted taken, predicted not taken or no static branch prediction is provided.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.